Muñoz Organero, Mario; Brito Pacheco, Claudia. Improving accuracy and simplifying training in fingerprinting-based indoor location algorithms at room level. *Mobile Information Systems*, 2016 (2682869), 16 pages.

DOI: https://doi.org/10.1155/2016/2682869

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 2682869, 16 pages
http://dx.doi.org/10.1155/2016/2682869

Research Article

Improving Accuracy and Simplifying Training in Fingerprinting-Based Indoor Location Algorithms at Room Level

Mario Muñoz-Organero and Claudia Brito-Pacheco

Department of Telematics Engineering, Universidad Carlos III de Madrid, 28911Leganes, Spain

Correspondence should be addressed to Mario Muñoz-Organero; munozm@it.uc3m.es

Received 9 August 2015;Revised 14 December 2015;Accepted 5 January 2016

Academic Editor: Francesco Gringoli

Fingerprinting-based algorithms are popular in indoor location systems based on mobile devices. Comparing the RSSI (Received Signal Strength Indicator) from different radio wave transmitters, such as Wi-Fi access points, with prerecorded fingerprints from located points (using different artificia intelligence algorithms), fingerprinting-based systems can locate unknown points with a few meters resolution. However, training the system with already located fi gerprints tends to be an expensive task both in time and in resources, especially if large areas are to be considered. Moreover, the decision algorithms tend to be of high memory and CPU consuming in such cases and so does the required time for obtaining the estimated location for a new fi gerprint. In this paper, we study, propose, and validate a way to select the locations for the training fingerprints which reduces the amount of required points while improving the accuracy of the algorithms when locating points at room level resolution. We present a comparison of different artificia intelligence decision algorithms and select those with better results. We do a comparison with other systems in the literature and draw conclusions about the improvements obtained in our proposal. Moreover, some techniques such as filtering nonstable access points for improving accuracy are introduced, studied, and validated.

1. Introduction

Indoor location systems provide a solution for user navigation in places where GPS signals are not available. There are many alternatives available for locating users indoors that can be categorized in several ways according to the mathematical algorithms used, radio technologies required, or hardware components used. Some of the alternatives are based on the user carrying a mobile device and others are based on device free [1]. One classical categorization of indoor location systems divides them into presmartphone and smartphone eras. The former are based on specific hardware components like infrared badges, ultrasound tags, laser rangers, and wireless modules such as RFID [2]. Since they require additional equipment to be carried by users, they tend to be difficult to deploy and use in real scenarios. The latter are based on the use of smartphones as the user device for indoor location [3, 4]. Smartphones are reliable and user friendly tools that allow people to have access to information services anytime and anywhere. Some of these services may require using the location of the user and, consequently, integrating indoor location services in smartphone based systems is a natural way to minimize deployment requirements. Smartphones have access to several radio technologies such as Wi-Fi, Bluetooth, or terrestrial cellular mobile networks. Th y incorporate built-in sensors such as accelerometer, gyroscope, magnetometer, microphone, or camera. They are, therefore, computing platforms which are capable of sensing the environment and applying the required calculations in order to locate the users in scenarios where the GPS signals are not available.

Smartphone based indoor location systems can be categorized into two major families: on-demand location systems and continuous user location tracking systems. The former normally rely on fingerprinting techniques [5–9] while the latter tend to use inertial embedded sensors to calculate incremental user displacements [10, 11].

This paper focuses on fingerprinting smartphone-based algorithms which are employed to locate users at room level (the exact location of the user inside a room is not that relevant but knowing the exact room the user is in with the highest accuracy is crucial). Using room level granularity will allow us to overcome one of the crucial aspects of indoor location systems when the areas to cover are big: scalability [12]. Training indoor locations systems with a complete grid of training points is time consuming and does not scale on computational resources required for estimating user locations. Some research proposals such as [12] overcome this limitation by using crowdsourcing-based place learning. However, this approach requires the manual involvement of volunteer collaborating users to improve the accuracy of the system and collateral aspects such as data quality should be taken into account. We will focus on optimizing the way by which to select sample points to train the system by considering the geometry of the areas in which to position users.

In summary, this paper contributes in 3 major ways:

(1) The way to select the most convenient sampling points to measure training fingerprints in order to increase the accuracy while minimizing the memory and CPU execution requirements. Th cost for training the system includes both the number of training points and the number of fingerprints at each point. The paper focuses on reducing the cost associated to the number of points not adding additional constrains on the number of fingerprints at each training point.

(2) The analysis and evaluation of the performance of different decision algorithms.

(3) The way to fi ter unstable access points in order to increase accuracy and minimize execution requirements.

The rest of the paper is organized as follows. Section 2 presents a brief review of the major algorithms that have been previously proposed in order to estimate the current user location based on already located fingerprints used when training the system. It also defines the scope of the research in this paper and its major contributions. In Section 3 we propose several options for deciding where to take fi gerprints in order to best train the system. Previously used alternatives are presented together with a novel mechanism that will be analysed and validated in Section 6. In Section 4 we present a mechanism designed to select the best WAPs among those available. This section shows how this mechanism provides optimal results in terms of the relative accuracy provided, as well as the computational resources needed. Noise and stability filters are presented. Section 5 introduces different approaches to combine several fi gerprints when locating unknown points in order to improve results. This is done thanks to the analysis of the time fluctuations of the received signal strengths. Section 6 captures the validation results using experimental data in two real case scenarios. Conclusions are presented in Section 7.

2. Fingerprinting-Based Algorithms

Several different approaches and technologies have been proposed, analysed, explored, and validated for indoor positioning systems, fingerprinting being the preferred alternative for smartphone-based indoor locations systems [13]. Many approaches have been tried on location fingerprinting [14]. *K*NN (*K*-Nearest Neighbor), which converts fingerprints into vectors and chooses the K historical fi gerprints with the shortest distance to the measured fingerprint to estimate the location, has been used in algorithms such as Redpin [14]. Authors in [15] provide a general comparison of SVM (support vector machines), *K*NN, Bayesian modeling, and multilayer perceptrons. Carlotto et al. [16] uses a statistical Gaussian Mixture Model. Authors in [14] propose an enhanced version of the Redpin algorithm by modifying the signific nce of each AP (access point) at each location according to their correlation and introduce a noise filter to discard less frequently visible APs at each location. Th authors compare their results with Naïve Bayes classifie s (NBC) and SVM based algorithms. Authors in [17] propose a Wi-Fi indoor location technology based on the combination of fingerprinting and the use of *K*-Means algorithm. Th diff rent approaches are normally compared using two major metrics: relative accuracy (%) and average errors (m) [13].

The authors in [18] present an analytical model for analyzing fingerprinting-based indoor location systems. Th y develop the framework which analyses a simple positioning system that employs the Euclidean distance between a sample signal vector and the location fingerprints of an area stored in a database. They analyse the effect of the number of access points that are visible and radio propagation parameters on the performance of the positioning system and provide some preliminary guidelines on its design. Th y predict that in optimal circumstances a resolution of a few meters could be expected in such systems. This predicted result for best algorithm's resolution has been a common fact for the latest developed systems [13]. Authors in [19] propose the idea of using Virtual Access Points (VAPs) in order to improve the location's accuracy. The idea is particularly significant if the number of available access points is small. The resolution of the systems using a *K*NN (*K*-Nearest Neighbors) approach is again of a few meters approximately. A recent experiment comparing 13 infrastructure-based and 9 infrastructure free indoor location technologies showed that best results for Wi-Fi fi gerprint based technologies were close to 1 meter when using Bayesian filters [20] but that the localization accuracy degrades as much as 3 meters due to setup and environmental changes, such as human or furniture changes.

Indoor location systems focus on solving the problem of best estimating the user location. Th exact user location inside a building is occasionally required. In some occasions, however, the location is only required to be estimated at room or area levels. Smartphone-based indoor location systems which rely on fingerprinting can be used for either findi g the best estimate for the user location inside a building or for findi g the best room or area in which the user is located. In the first case it was found that best estimate means that an exhaustive training of the system is required in order to

measure fingerprints in as many training locations as possible. Trying to estimate the user location at a room level involves, on the other hand, different training requirements for the system. In this paper we focus on best solving this second problem. In the next section we analyse different alternatives for selecting the best training locations for indoor location fingerprinting-based algorithms. The validation results are presented in Section 6.

Th major contribution of this paper is the study, analysis, and evaluation of the relative accuracy of the different smartphone fi gerprinting-based indoor location algorithms when locating users at room level. This has been done using different strategies for selecting the points used to train the system, applying different values for noise and stability filters in order to select the best WAPs, and taking the average of several measured fingerprints when locating the user. In particular the following algorithms have been used:

(i) ADTree, BFTree, Decision Stump, FT, J-48, J-48graft, LADTree, LMT, NBTree, Random Forest, Random Tree, REPTree, SVM, Naïve Bayes, *K*NN, Redpin, and WASP.

Th confidenc is also analysed in order to select the best algorithms, the number of WAPs to be considered, the values for the noise and stability filters, and the location of the training points. The results are also validated by comparing them to those presented in papers which deal with similar aspects.

3. Selecting Training Points

Smartphone finge printing-based indoor location systems require a training phase to be carried out before they can be used to locate new unknown points. Select the locations where taking measures for the set of training fi gerprints is by itself a decision that has to be studied, analysed, and optimized taking into account both the fin l relative accuracy of the location algorithm and the underlying computational requirements. Two major approaches are examined in previous studies: using a complete grid of points which cover the entire part of the building where users are to be located [18] and using a randomly distributed set of points inside the building [14] which cover the entire area but without the restriction of having to use a regular grid of training locations. The first approach aims to provide a good resolution for the estimated location of unknown points in every part of the building by considering a number of training fi gerprints that grows as d^3 (being d the linear dimension of the building). Th amount of time required to take the training fingerprints and the computational cost in terms of memory, CPU, and execution time needed to run the location algorithms are major drawbacks of this approach if big areas are to be covered. Randomly selecting the locations to measure the training fingerprints allows us to provide better resolutions in certain areas of interest and to relieve the requirements on the total amount of fingerprints to be measured. This approach has been used in [14] in order to provide location estimates at room level for certain rooms in a building. The major problem of this approach is that it does not give us a mechanism to optimize the number of points to consider for training the system but a mechanism which simply selects them in the areas of interest.

In this paper, we propose a third approach to minimize the requirements for the number of training fingerprints when using the indoor location algorithms at room level. We propose to only measure the points in the border of each room without fingerprinting the interior part of each area. This approach is aligned with the objective of the system to be designed: to differentiate among rooms without taking into account user movements inside each room. The results in Section 6 show how this approach can provide even better accuracy than exhaustive fingerprinting of each room with a smaller number of training fi gerprints required.

One of the parameters to be considered when fingerprinting an area for training the indoor location system is the distance between adjacent points that need measuring. Th authors in [18] propose that in order to select distance between the points for optimal sampling for training fingerprints, the compromise between granularity and accuracy should be taken into account. Th y do not recommend selecting points which are less than 1 meter away from each other. Th review done in [13] of the different algorithms used for locating users indoors also shows that optimal resolutions provide an average error of several meters. The Redpin algorithm, for example, provides an average error of 3 meters. Experimental results in Section 6 are carried out taking these results into consideration.

4. Applying Noise and Stability Filters

The temporal stability of the RSSI has a major impact on the results of the algorithms that use it to estimate the location of the user in indoor environments. Th authors in [18], when presenting an analytical model for analyzing fingerprinting-based indoor location systems, recognise the impact that the value of the variance of the measured RSSIs has on predicted results. While they develop a model to predict the expected accuracy of a fi gerprinting-based system based on the measured parameters, they do not propose mechanisms to improve the accuracy by introducing additional factors to improve nonmodifi ble parameters such as the standard deviation of signal strength over time from a single access point. In this section we propose the application of noise and stability filters to improve results when unpredictable and unavoidable changes in the physical setting have to be considered.

Authors in [14] implement a noise filter to remove less frequently visible APs for a given location, considering these signals as noise. They divide the APs into relevant and noisy APs. Th y also take into account the percentage of times in which relevant APs contribute to the fingerprints in one location and create a correlation index between locations and APs. The obtained results improve those of the basic algorithm and constitute a promising fiel in order to further optimize the relative accuracy of fingerprinting-based indoor location systems.

In this paper we propose (and validate in Section 6) an improvement of the mechanisms proposed in [14] to reduce the impact of the fluctuations of the measured RSSIs on

the location estimation results. We propose to fi ter the WAPs performing a stability based classific tion using different thresholds. The more restrictive the threshold is, the more stable the WAPs to be selected are (those with measured RSSIs fluctuating less over time). However, a more restricted threshold also implies a smaller number of WAPs to be selected which may have a negative impact on the results. The results presented in [14] show that a number of 10–15 WAPs is enough to obtain optimal results. The algorithm in which we propose to select this set of optimal WAPs in terms of stability is the following:

(1) Take several measures over time in the same location to train the system (ideally on different days and different times).

(2) Mark as potentially unstable those WAPs whose number of noisy RSSIs is, for a particular training location, below a noise threshold.

(3) Mark as potentially unstable those WAPs whose variance for the RSSI is, for a particular training location, greater than a stability threshold.

(4) Finally, those WAPs marked as potentially unstable must be removed if the number of training points where they are marked as potentially unstable is above a threshold.

Th selection of the values of these thresholds will provide a mechanism to select either a smaller number of more stable WAPs or a bigger number of less stable WAPs. The results captured in Section 6 show that selecting the best 10–16 WAPs will provide the optimal results.

In more detail, the following four thresholds have been set for the analysis performed in this paper.

(1) Noise Threshold (NT). We set its value to −80 dBm as it is considered the minimum acceptable RSSI value for establishing a connection.

(2) Number of Noisy Samples Threshold (NNST). It is calculated according to the following formula:

$$\text{NNST} = n\text{Samples} * \text{factor}, \text{ where } n\text{Samples} \in \aleph^*, \text{factor} \in [0 \cdots 1] \text{ and } (n\text{Samples} * \text{factor}) \in \aleph^*.$$

(1) Variance Threshold (VT). We set its value to 60 dBm^2 (~8 dBm). This means that we choose ±4 dBm as the acceptable range of oscillation of a set of RSSI values for a given WAP. Above this threshold we consider that it is not possible to achieve stable transfer rates.

(2) Location Threshold (LT). It is calculated according to the following formula:

$$\text{LT} = n\text{Locations} * \text{factor}, \text{ where } n\text{Locations} \in \aleph^*, \text{factor} \in [0 \cdots 1] \text{ and } (n\text{Locations} * \text{factor}) \in \aleph^*.$$

Having established the above thresholds, the first step of the algorithm is to calculate the number of noisy samples by WAP for each location. A sample is considered a noisy one when its RSSI is less than the value set as NT (note that the WAP is not considered noisy if it is not detected at a certain point, only if it is detected with a fluctuating power below the noise threshold a number of times above the NNST).

Then for each location and wireless access point we check if the result obtained in the previous step equals or exceeds the NNST. If so, the WAP is marked as unstable for that location. If not, the variance of its RSSIs is calculated. If this number exceeds the VT, the WAP is marked as unstable for that location. In summary, we consider that a WAP is unstable at a given location when the number of noisy samples for this access point in that location equals or exceeds the NNST or the variance of the RSSIs for that wireless access point and location exceeds the VT.

Finally, for each wireless access point we add the number of locations where the WAP is considered unstable. If the result equals or exceeds the LT we eliminate this WAP.

Depending on how the previous four thresholds (noise, number of noisy samples, variance, and location) are configured we can be more or less strict when taking into account a smaller or larger number of WAPs. For instance, if we keep fixed NT and VT, we take a number of samples (*n*Samples) and locations (*n*Locations) equal to 10 and 12, respectively, and adjust the factors involved in the calculation of NNST and LT; we would obtain the results shown in Table 1.

Table 1: Configur tion of UMR and UL thresholds.

#WAPs	5	8	10	16	20	32
			NT = −80 dBm			
			*n*Samples = 10			
Factor	0.7	0.7	0.7	1	1	1
NNST	≥7	≥7	≥7	≥10	≥10	≥10
			VT = 60 dBm^2 (~8 dBm)			
			*n*Locations = 12			
Factor	0.5	0.6~	0.75	0.75	0.83~	1
LT	≥6	≥8	≥9	≥9	≥10	≥12

In the first row of Table 1, the number of wireless access points that would result after applying our algorithm to delete unstable WAPs, with the configur tions indicated for NNST and LT, is shown. As can be appreciated, as we expand both thresholds we are less strict; hence, we obtain a larger number of WAPs.

5. Optimizing the Location Estimations by Fingerprint Averaging

Another mechanism that may be used to improve the performance of the fingerprint based indoor location algorithms is combining several fi gerprints measured when trying to estimate the location of an unknown point. Th more the fingerprints over a longer period of time the better the impact one expects to obtain in the predicted results. However, there is a trade-off in the number of fingerprints that can be taken into account in a real setting since the user expects to obtain a result from the algorithm in a reasonable amount of time. In this paper we will analyse the impact on the relative accuracy

Figur e 1: Experiments environment, as seen from our Android training application (Google Maps API v2).

obtained when combining several consecutive fingerprints taken over a few seconds interval. In order to combine the fi gerprints we propose two different algorithms (the results will be presented in Section 6):

(1) Calculate the average of the measured fingerprints and make an estimate of the location of this average fi gerprint.

(2) Estimate the location of each of the measured fingerprints and make a majority based decision in order to select the most likely location.

Section 6 shows the results when 2, 3, or 5 finge prints are used in the average.

6. Experimental Results

6.1. Environment. All the tests described in this paper took place in the first floor of the Torres Quevedo Building (including the hallway and the research lab 4.1.C.03) of the Telematics Engineering Department at the Carlos III University of Madrid. In Figure 1, the map corresponding to the interior of the first floor of the building is shown and the location of the selected points for the two experiments conducted is presented.

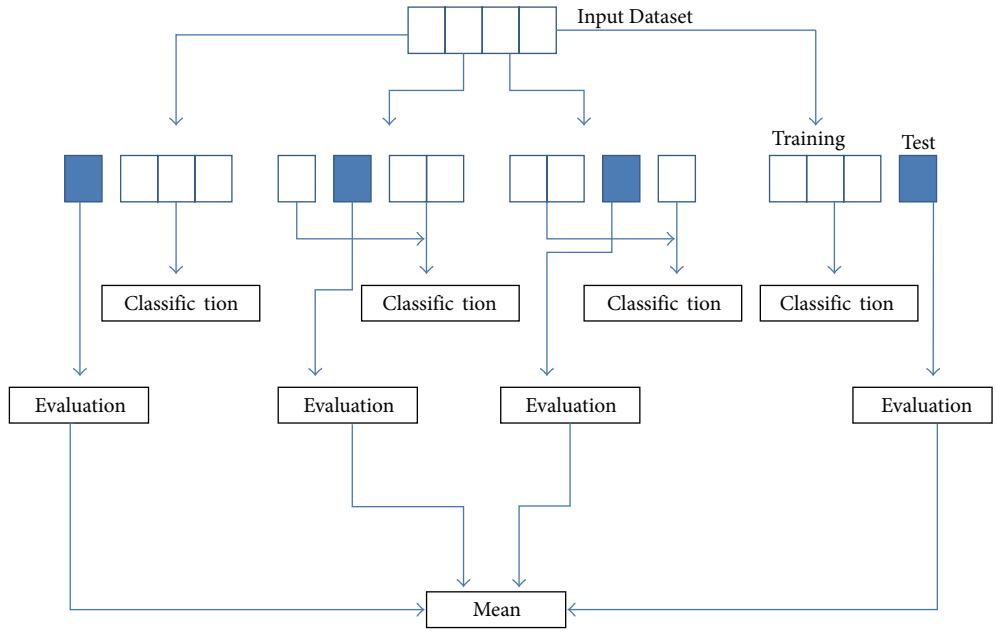
For the fi st experiment 12 post-its (4 for each area) were arranged as training labels and they were separated by a distance of 3 meters. These markers indicated the points that were going to carry out the sampling. In total there were 12 training locations that we grouped and divided into 3 sequential areas: 6 interior area locations and the other remaining 6 were the ones we called "border" locations and we defin d the three areas designated on the map. For each of the locations we needed to make 5 consecutive measurements (at intervals of 15 s) of the RSSI of Wi-Fi networks. Once the 12 points had been scanned we repeated the same procedure for each location; hence the training had involved 10 steps in two batches of 5 spaced in time. This made a total of 120 (12 locations $*$ 10 measures) fingerprints. The next day we performed the same experiment, which made it in total a final set of 240 fi gerprints.

For the second experiment, a more complex setting was selected. We used 3 areas physically separated by walls and doors (two connected aisle sections, separated by a connecting door and a research laboratory wall to wall with one of the aisle sections). We took 6 location points in each of these areas following the proposed approach to sample points in the border between areas. The borders sample points have been measured with a 2-3-meter separation criteria.

6.2. Experimental Methods. To carry out the analysis of the tests we used 10-fold cross-validation to estimate how accurate a model is before implementing it. Using this technique the input data set is divided into 10 equal-sized subsets. Of these, one subset is taken to validate the generated model, while the remaining nine are used to train the system and generate the corresponding model. This process is repeated 10 times, using each of the 10 subsets exactly once to validate the model. Th 10 results from the 10 iterations are then averaged to produce a single estimation. The sampling type we have chosen is "stratifi d" because not only does it generate random subsets but also ensures that the class distribution in these subsets is the same as in the initial set (Figure 2).

In all experiments we simulated, using the RapidMiner Studio 6.0.008 tool, what would be the response of our indoor location system when training with each of the following algorithms?

(1) ADTree. Th basic algorithm is based on [21]. Thi current version only supports two-class problems. The number of boosting iterations needs to be manually tuned to suit the dataset and the desired complexity/accuracy trade-off. The trees' induction has been optimized, and heuristic search methods have been introduced to speed learning.



Figur e 2: K-fold cross-validation scheme, with $K = 4$ and one classifie .

(2) BFTree. Thi algorithm uses binary split for both nominal and numeric attributes. For missing values, the method of "fractional" instances is used. For more information, see [22].

(3) Decision Stump. Thi algorithm is usually used in conjunction with a boosting algorithm. It does regression (based on mean-squared error) or classification (based on entropy). Missing is treated as a separate value.

(4) FT. Functional Trees are classific tion trees that could have logistic regression functions at the inner nodes and/or leaves. The algorithm can deal with binary and multiclass target variables, numeric and nominal attributes, and missing values. For more information, see [23, 24].

(5) J-48. We use this algorithm to generate a pruned or unpruned C4.5 decision tree. For more information, see [25].

(6) J-48graft Thi algorithm generates a grafted (pruned or unpruned) C4.5 decision tree. For more information, see [26].

(7) LADTree. We use this algorithm to generate a multiclass alternating decision tree using the LogitBoost strategy. For more information, see [27].

(8) LMT. Logistic Model Trees are classific tion trees with logistic regression functions in the leaves. The algorithm can deal with binary and multiclass target variables, numeric and nominal attributes, and missing values. For more information, see [28, 29].

(9) NBTree. Thi algorithm generates a decision tree with Naive Bayes classifie s in the leaves. For more information, see [30].

(10) Random Forest. We use this algorithm to generate a forest of random trees. For more information, see [31].

(11)Random Tree. Thi algorithm generates a tree that considers K randomly chosen attributes at each node, performs no pruning, and also has the option to allow estimation of class probabilities based on a hold-out set (backfitting).

(12) REPTree (Fast Decision Tree Learner). It builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning (with backfitting) and only sorts values for numeric attributes once. Missing values are dealt with by splitting the corresponding instances into pieces (i.e., as in C4.5).

(13) SVM. Thi learner uses the Java implementation of the support vector machine mySVM by Stefan Rueping. Thi learning method can be used for both regression and classific tion and provides a fast algorithm and good results for many learning tasks.

(14) KNN ($K = 5$). Th K-Nearest Neighbor algorithm is based on learning by analogy, that is, by comparing a given test example with training examples (K-Nearest Neighbors) that are similar to it. "Closeness" is defin d in terms of a distance metric, such as the Euclidean distance.

(15) Naïve Bayes. A Naïve Bayes classifie is a simple probabilistic classifie based on applying Bayes' theorem (from Bayesian statistics) with strong (naive) independence assumptions. In simple terms, a Naive Bayes classifie assumes that the presence (or absence) of a class' particular feature (i.e., attribute) is unrelated to the presence (or absence) of any other feature.

(16) Redpin. This algorithm uses a weighted combination of the vector distance and the AP similarity and chooses $K = 1$

Figure 3: "Border" locations for each area.

to decide the best match. The default configur tion is $\alpha = 1$, $\beta = 0.4$, and $\gamma = 0.2$ and the value of C function is within the range −1 to 1, depending on the contribution of the signal strength of two common Aps. The Redpin source code can be found here [32].

(17) WASP. Thi algorithm uses Point-Wise Mutual Information (PMI) as the weight for one AP to one location and chooses $K = 5$ to decide the best match. It also applied noise filter to remove irrelevant Aps.

6.3. Results. Thi section captures the results of both experiments carried out. Section 6.3.1 presents the results of the simpler scenario based on the measurements on a single aisle section (divided into 3 areas) and Section 6.3.2 captures the results of the second experiment based on measurements from 3 different wall and door separated areas.

6.3.1. Results from the Linear Configuration. In this section we present a comparison of the algorithms described in Section 6.2 for the case of the linear configur tion in the aisle of the building. Th metric we used to evaluate them is called "relative accuracy" and is commonly used in fingerprinting-based systems. In our case we aim to measure the number of test fi gerprints which are correctly classifi d according to the area to which they belong.

The e algorithms receive as input parameters the area in which the fi gerprint was taken and the RSSIs values measured for each WAP. Initially we started with a set of 40 WAPs which involved training the system with 40 (RSSIs for each WAP) + 1 (area) parameters. Our fi st objective was using our algorithm to delete unstable WAPs (previously described in Section 4), discard those access points that do not provide any relevant information to train the algorithms, and verify that the results improved compared to when the process was performed without fi tering.

Our second challenge was to study the behaviour of the algorithms when we trained the system with only border locations, that is, the ones that delimit an area, and to analyse whether the results were better than when all locations were used (or at least comparable, since training the system only with border locations simplifie the training face and increases scalability). In Figure 3, each geometric shape represents an area of the hallway in which the tests were conducted. "Border" locations are the ones that look darker.

Finally, we wanted to analyze how the variations in the measured RSSIs values overnight affect the predictions made by the system and if there was any way to improve them when applying our noise and stability filters (in our case, the building is dedicated to research activities involving wireless networks and some of the WAPs were moved of switched off from one day to the other).

In the following two subsections we show a comparison of the results we obtained when training the algorithms with all locations (12) compared to those obtained when we only use the "border" (6) and validate the models generated considering all locations (interior and borders). These results are presented in conjunction with the ones coming from our algorithm to delete unstable WAPs. This way we get a better overview of the impact of our three major contributions to improve the problem of indoor location based on smartphones and finge printing and for the metric "relative accuracy."

Table 2: Relative accuracy by algorithm based on the number of WAPs when training with all locations (part I).

#WAPs	ADTree	BFTree	Decision Stump	FT	J-48
40	95.83 ± 5.59	95.00 ± 4.08	95.00 ± 5.53	97.50 ± 3.82	95.83 ± 5.59
5	85.83 ± 9.17	85.83 ± 10.57	86.67 ± 6.67	79.17 ± 11.33	87.50 ± 10.03
8	90.00 ± 8.16	88.33 ± 8.50	87.50 ± 10.03	88.33 ± 13.02	85.83 ± 7.50
10	**96.67 ± 4.08**	**95.00 ± 6.67**	**92.50 ± 7.86**	**97.50 ± 3.82**	**93.33 ± 6.24**
16	95.83 ± 4.17	95.83 ± 5.59	92.50 ± 7.86	96.67 ± 5.53	93.33 ± 6.24
20	95.83 ± 5.59	94.17 ± 5.34	92.50 ± 7.86	97.50 ± 3.82	93.33 ± 6.24
32	95.83 ± 5.59	94.17 ± 5.34	92.50 ± 7.86	95.83 ± 5.59	94.17 ± 5.34

(a) Training the System with All Locations. In Tables 2–4 we show the results we obtained in the first row for the different algorithms when we train the system with all locations (12) and we have not executed our algorithm to delete unstable WAPs. In this case we choose all the WAPs from the initial set (40). The we show the values obtained when we also train the system with all locations (12) but this time we run our algorithm with the configur tion previously detailed in Table 1 to take 5, 8, 10, 16, 20, and 32 access points, respectively.

Comparing the results for the first and fourth rows we can see that out of the 15 algorithms evaluated, the results are maintained or improved significantly for 9 of them (ADTree[+0.84], BFTree, FT, LADTree[+5.84], Random Forest, Random Tree[+3.34], REPTree[+0.84], *K*NN[+1.66], and NBC[+11.67])and slightly worse for the remaining 6 (Decision Stump[−2.5], J-48[−2.5], J-48graft −0.83], LMT[−1.66], NBTree[−0.83], and SVM[−3.34]). Although this decline

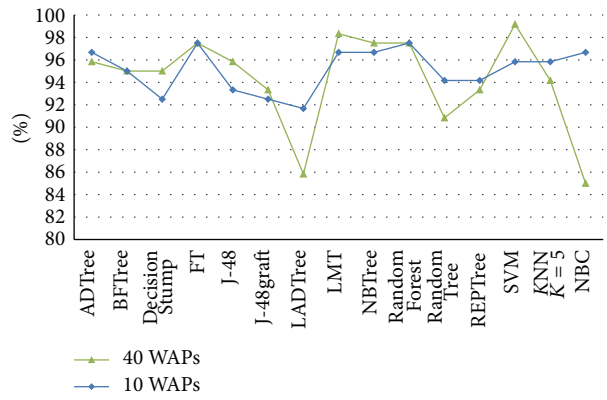


Figure 4: Relative accuracy by algorithm as we take 40 or 10 WAPs.

Table 3: Relative accuracy by algorithm based on the number of WAPs when training with all locations (part II).

#WAPs	J-48graft	LADTree	LMT	NBTree	Random Forest
40	93.33 ± 5.00	85.83 ± 12.94	98.33 ± 3.33	97.50 ± 5.34	97.50 ± 3.82
5	86.67 ± 10.00	88.33 ± 7.64	77.50 ± 11.21	85.00 ± 8.98	85.83 ± 9.17
8	85.83 ± 7.50	87.50 ± 9.32	87.50 ± 7.68	91.67 ± 9.13	92.50 ± 5.83
10	**92.50 ± 7.86**	**91.67 ± 8.33**	**96.67 ± 4.08**	**96.67 ± 4.08**	**97.50 ± 3.82**
16	91.67 ± 7.45	90.83 ± 10.17	96.67 ± 5.53	98.33 ± 3.33	96.67 ± 4.08
20	91.67 ± 7.45	90.00 ± 11.67	96.67 ± 5.53	95.00 ± 5.53	97.50 ± 3.82
32	91.67 ± 7.45	90.83 ± 10.17	95.00 ± 5.53	98.33 ± 3.33	98.33 ± 3.33

Table 4: Relative accuracy by algorithm based on the number of WAPs when training with all locations (part III).

#WAPs	Random Tree	REPTree	SVM	KNN K = 5	NBC
40	90.83 ± 7.86	93.33 ± 6.24	99.17 ± 2.50	94.17 ± 6.51	85.00 ± 7.26
5	80.83 ± 10.57	82.50 ± 7.86	71.67 ± 10.00	83.33 ± 9.86	78.33 ± 9.28
8	89.17 ± 8.37	86.67 ± 11.90	85.00 ± 11.06	87.50 ± 11.33	79.17 ± 11.93
10	**94.17 ± 6.51**	**94.17 ± 7.50**	**95.83 ± 5.59**	**95.83 ± 5.59**	**96.67 ± 4.08**
16	92.50 ± 8.70	93.33 ± 8.16	97.50 ± 5.34	95.83 ± 5.59	97.50 ± 3.82
20	90.00 ± 10.41	92.50 ± 9.46	98.33 ± 5.00	96.67 ± 5.53	95.00 ± 6.67
32	87.50 ± 9.32	93.33 ± 8.98	98.33 ± 3.33	95.83 ± 5.59	86.67 ± 11.30

reaches a peak of 3.34 percentage points for the algorithm of support vector machines (SVM), this number seems negligible since we generally get better results. At the same time we are reducing the computational cost substantially, going from 40 to 10 WAPs.

When we compare the results published for the WASP and Redpin algorithms with noise filtering and 5 access points [11] with the ones we get when we run our algorithm to delete unstable WAPs (and also take 5 WAPs) we can see that we get significant improvements for the following algorithms: ADTree, BFTree, Decision Stump, J-48, J-48graft LADTree, NBTree, Random Forest, REPTree, and *K*NN. If we select the algorithm which provides the best resolution with 5 WAPs (LADTree [88.33]) we see that the difference from Redpin and WASP is +6.33 percentage points. For 8 access points we managed to improve the results very signific ntly with NBTree (91.67) and Random Forest (92.50) compared to results published for WASP with 8 APs [33]. Thi is an improvement of +2.50 percentage points when using the Random Forest algorithm.

To generate the comparative chart on Figure 4 we have considered the case where our algorithm to delete unstable WAPs gives us 10 access points since for a larger number we did not observe a signific nt improvement. We have used the green colour to show the results when we have not run our algorithm and thus the number of input parameters that the algorithms receives is equal to the RSSIs corresponding to the initial number of access points (40) plus the area. In blue we show the results we get when we have executed our algorithm with a given configur tion to take 10 access points.

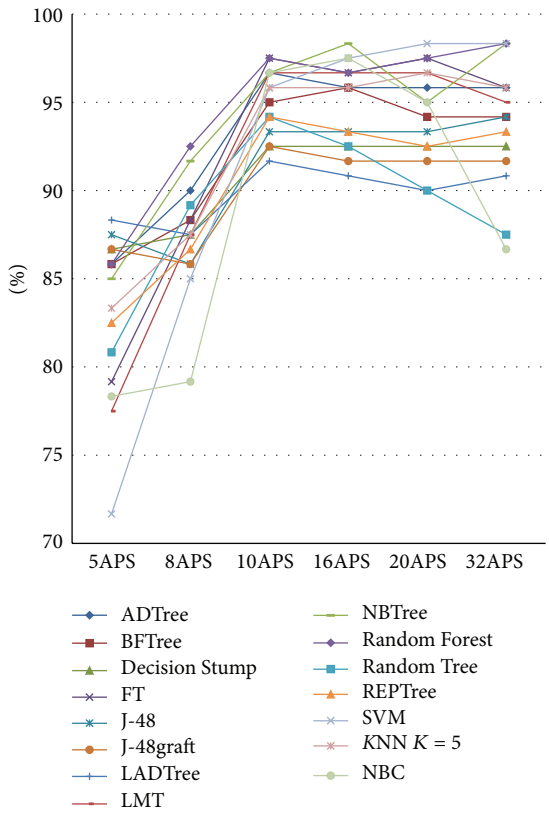


Figure 5: Relative accuracy by algorithm based on the number of WAPs.

In this case the number of parameters that will be passed to the algorithms will be equal to the RSSIs corresponding to the 10 WAPs selected plus the area (30 parameters less than the previous case).

Figure 5 shows a comparison of the "relative accuracy" for algorithms evaluated when the system is trained with all locations and by the number of WAPs selected. For 8 access points the algorithm that provides the best resolution is Random Forest while for 10 the FT and Random Forest algorithms are at the same level.

(b) Training the System with "Border" Locations. Th results of this second test are presented in the same format as those already discussed in the previous section. The only difference is that now the system has been trained with only half of the locations (6), those we have called "border," and has been validated with all (internal and border) (Tables 5, 6, and 7).

If we compare the values and reliability intervals obtained in this second test for the metric "relative accuracy" we can see a significant improvement as compared to when all locations were used. Th increases/decreases in percentage points in those cases where we did not execute our algorithm to delete unstable WAPs are as follows (we take all the WAPs from the initial set, as indicated in the fi st row of combined Tables 2–4 and combined Tables 5–7): ADTree[+2.34], BFTree[+0.17], Decision Stump[−0.50], FT[+2.33], J-48[−4.33], J-48graft[−1.16], LADTree[+2.17], LMT[+0.67], NBTree[+0.17], Random Forest[+1.50], Random Tree[+0.67], REPTree[−0.83], SVM[−1.17], *K*NN[+2.83], and NBC[+2.17].

Although for five (Decision Stump, J-48, J-48graft, REPTree, and SVM) of the fi een algorithms evaluated the percentages decrease, when we run our algorithm to rule out unstable WAPs these results improve substantially (Decision Stump[+4.17], J-48[+2.50], J-48graft[+3.33], and SVM[+2.34]). Just to REPTree [−0.84] we do not get an improvement. For other algorithms markers would be as follows: ADTree[+2.33], BFTree[+1.83], FT[+1.67], LADTree[+0.16], LMT[+1.16], NBTree[+1.00], Random Forest[+1.00], Random Tree[+3.16], *K*NN[−3.66], and NBC[+0.33].

Comparing the results of the second test for the firs and fourth rows we can see that out of the 15 algorithms tested, the results are maintained or improved signific ntly for 11 of them (ADTree[+0.83], BFTree[+1.66], Decision Stump[+2.17], J-48[+4.33], J-48graft[+3.66], LADTree[+3.83], NBTree, Random Tree[+5.83], REPTree[+0.83], SVM[+0.17], and NBC[+9.83]) and slightly worse for the remaining 4 (FT[−0.66], LMT[−1.17], Random Forest[−0.50], and *K*NN[−4.83]). Although this decline reaches a peak of 4.83 percentage points for the algorithm of *K*-Nearest Neighbors, this number seems negligible since generally we get better results, especially for NBC and Random Tree algorithms. At the same time, again, we are reducing the computational cost substantially, going from 40 to 10 WAPs.

Table 5: Relative accuracy by algorithm based on the number of WAPs when training with border locations (part I).

#WAPs	ADTree	BFTree	Decision Stump	FT	J-48
40	98.17 ± 0.90	95.17 ± 1.57	94.50 ± 3.95	99.83 ± 0.50	91.50 ± 4.31
5	91.33 ± 1.94	87.67 ± 2.13	85.50 ± 1.07	70.50 ± 5.00	85.00 ± 3.07
8	93.33 ± 0.75	94.17 ± 0.83	89.83 ± 0.50	88.17 ± 4.74	90.17 ± 2.63
10	**99.00 ± 1.11**	**96.83 ± 0.50**	**96.67 ± 0.00**	**99.17 ± 0.83**	**95.83 ± 1.34**
16	97.33 ± 1.33	95.50 ± 0.76	94.83 ± 0.50	99.17 ± 1.12	95.17 ± 1.57
20	97.50 ± 1.12	95.83 ± 1.34	94.83 ± 0.50	98.83 ± 1.07	96.17 ± 1.67
32	98.00 ± 1.63	94.17 ± 2.71	94.83 ± 3.61	99.00 ± 1.33	91.50 ± 3.02

Table 6: Relative accuracy by algorithm based on the number of WAPs when training with border locations (part II).

#WAPs	J-48graft	LADTree	LMT	NBTree	Random Forest
40	92.17 ± 3.73	88.00 ± 4.00	99.00 ± 1.11	97.67 ± 1.53	99.00 ± 0.82
5	85.33 ± 3.48	73.33 ± 15.26	77.83 ± 4.02	91.00 ± 2.71	90.00 ± 1.83
8	90.17 ± 2.63	91.50 ± 0.50	93.67 ± 1.63	92.00 ± 2.08	94.50 ± 0.76
10	**95.83 ± 1.71**	**91.83 ± 2.17**	**97.83 ± 1.67**	**97.67 ± 1.33**	**98.50 ± 1.17**
16	**95.17 ± 1.74**	**92.33 ± 2.13**	**98.33 ± 1.83**	**98.83 ± 1.30**	**99.00 ± 0.82**
20	96.17 ± 1.67	93.50 ± 2.52	98.83 ± 1.07	99.67 ± 0.67	98.33 ± 1.29
32	92.17 ± 2.59	87.33 ± 4.03	99.33 ± 1.11	98.00 ± 1.63	99.00 ± 0.82

When we compare the results published for the WASP and Redpin algorithms with noise filtering and 5 access points [11] with the ones we get when we run our algorithm to delete unstable WAPs (and also take 5 WAPs) we can see that we get signific nt improvements for the following

Table 7: Relative accuracy by algorithm based on the number of WAPs when training with border locations (part III).

#WAPs	Random Tree	REP Tree	SVM	*K*NN $K = 5$	NBC
40	91.50 ± 3.76	92.50 ± 3.59	98.00 ± 1.00	97.00 ± 1.63	87.17 ± 1.83
5	88.33 ± 1.29	82.50 ± 4.17	68.17 ± 1.74	80.00 ± 2.98	80.50 ± 1.07
8	87.83 ± 1.83	92.17 ± 2.89	87.50 ± 2.71	86.33 ± 1.94	81.17 ± 1.07
10	**97.33 ± 1.53**	**93.33 ± 3.07**	**98.17 ± 0.50**	**92.17 ± 1.30**	**97.00 ± 0.67**
16	95.67 ± 3.00	92.83 ± 3.66	98.33 ± 0.75	95.17 ± 1.89	96.67 ± 1.05
20	93.83 ± 3.66	93.83 ± 3.95	97.50 ± 0.83	94.67 ± 1.63	95.50 ± 1.30
32	95.83 ± 4.43	92.83 ± 3.66	99.33 ± 0.82	98.50 ± 0.90	87.50 ± 1.54

algorithms: ADTree, BFTree, Decision Stump, J-48, J-48graft, NBTree, Random Forest, Random Tree, and REPTree. If we select the algorithm which offers the best resolution with 5 WAPs (ADTree [91.33]) we observe that the difference from Redpin and WASP is +9.33 percentage points. For 8 access points we managed to improve the results very signific ntly with Random Forest (94.50) and BFTree (94.17) compared to results published for WASP with 8 APs [12]. This is an improvement of +4.50 percentage points when using the Random Forest algorithm.

Figure 6 compares the results for the metric "relative accuracy" when we run our algorithm to delete unstable WAPs (shown in blue) versus when we do not (shown in green). For most algorithms (11/15)percentages are maintained or improved signific ntly.

Figure 7 presents a comparison of the "relative accuracy" for algorithms evaluated when the system is trained only with "border" locations and by the number of WAPs selected. For 8 access points the algorithm that provides the best resolution is Random Forest while for 10 the ADTree and FT algorithms ranked first and second, respectively.

(c) Average Test Fingerprint to Validate the System Trained with "Border" Locations. Out of the 15 evaluated algorithms to conduct for this test we chose Random Forest, as this one gave us the best results overall for 16(99.00), 10(98.50), and 5(90.00) WAPs, respectively, as it is reflected in the previous paragraph "Training the System with 'Border' Locations."

Our goal here was to evaluate the system's predictions for each of the three areas we have trained and labeled "1" (left side hallway), "2" (central hallway), and "3" (right side

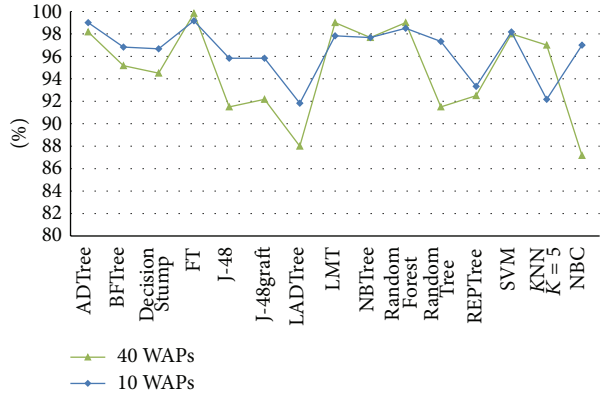


Figure 6: Relative accuracy by algorithm as we take 40 or 10 WAPs.

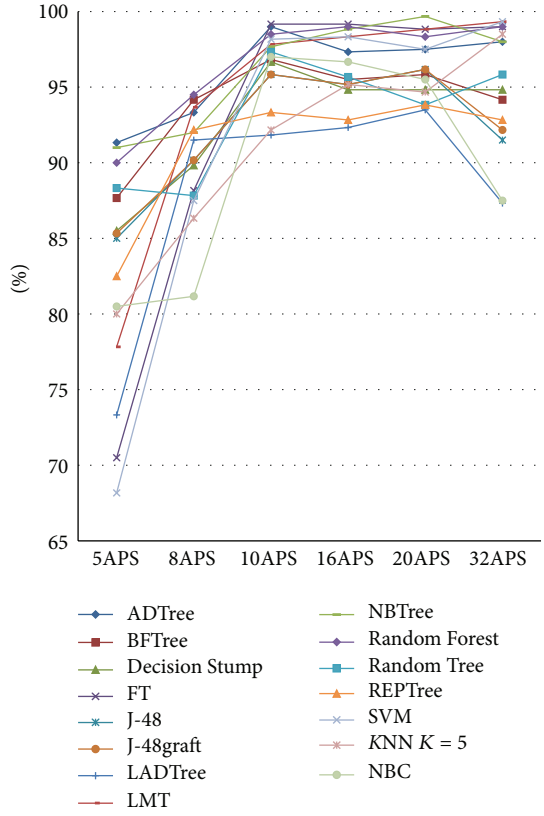


Figure 7: Relative accuracy by algorithm based on the number of WAPs.

hallway). As detailed at the beginning of Section 6 for each area we carried out measurements on two consecutive days in 4 different locations and for each we took 10 fi gerprints. Of these 4 locations we choose 2 (only for this test's purposes, as we are only taking into account "border" locations). This makes a total of 20 fingerprints (10 fingerprints ∗ 2 locations) for each area. Thus we have a set of training data of 60 fingerprints (20 fingerprints/area ∗ 3 area) per day. To validate the system, we selected as test data set both fingerprints (the ones corresponding to the day when we trained the system and the ones corresponding to the previous/following day) for all locations (indoors and borders). Here we only show the results corresponding to when we train the system with the fi gerprints that are taken on the second day and validated with the previous one. As we discussed at the beginning of Section 6.3, we intend to study how changes in the stability of the WAPs from one day to another affect to the predictions of the areas. Also, we aim to study how to adjust our algorithm to delete unstable WAPs with the configuration that we saw in Table 1 such predictions evolve as we take 16, 10, and 5 WAPs.

In Tables 8–13, we have the following: in the first row we point out the number of fingerprints, followed by the corresponding area identifie , the area forecast that the system gives us, and the reliability intervals for area "1," "2," and "3."

Table 8: System predictions for area 1 when using 16 WAPs (part I).

1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
0.6	0.6	0.6	0.6	0.7	0.8	0.8	0.6	0.5	0.7
0.3	0.2	0.3	0.3	0.2	0.0	0.1	0.2	0.2	0.0
0.1	0.2	0.1	0.1	0.1	0.2	0.1	0.2	0.3	0.3

Table 9: System predictions for area 1 when using 16 WAPs (part II).

11	12	13	14	15	16	17	18	19	20
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	**3**
0.5	0.7	0.9	0.7	0.7	0.4	0.6	0.5	0.4	0.4
0.3	0.0	0.0	0.2	0.2	0.4	0.2	0.3	0.3	0.1
0.2	0.3	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.5

Table 10: System predictions for area 2 when using 16 WAPs (part I).

21	22	23	24	25	26	27	28	29	30
2	2	2	2	2	2	2	2	2	2
2	2	2	**1**	2	2	2	2	2	2
0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.4	0.7	0.4	0.8	1.0	0.8	0.9	0.9	0.9
0.0	0.3	0.2	0.2	0.2	0.0	0.1	0.0	0.0	0.0

Table 11: System predictions for area 2 when using 16 WAPs (part II).

31	32	33	34	35	36	37	38	39	40
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	**3**	2
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	1.0	0.8	0.6	0.7	1.0	0.8	0.8	0.3	0.8
0.0	0.0	0.0	0.3	0.2	0.0	0.0	0.1	0.5	0.0

Table 12: System predictions for area 3 when using 16 WAPs (part I).

41	42	43	44	45	46	47	48	49	50
3	3	3	3	3	3	3	3	3	3
3	**2**	**2**	3	**2**	3	**2**	3	3	3
0.1	0.1	0.0	0.2	0.1	0.3	0.1	0.3	0.1	0.1
0.4	0.5	0.7	0.2	0.5	0.3	0.5	0.3	0.2	0.2
0.5	0.4	0.3	0.6	0.4	0.4	0.4	0.4	0.7	0.7

When we train the system with the fingerprints collected on the second day and validate with the ones corresponding to the first, adjusting our algorithm to take 16 WAPs, we obtain a relative accuracy of 80% (48/60 successes and 12/60 failures) (Tables 14, 15, 16, and 17).

For 10 access points the results are as follows: area 1 (20/20 successes and 0/20 failures) and area 2 (18/20 successes and 2/20 failures) (Tables 18, 19, 20, 21, 22, and 23).

Table 13: System predictions for area 3 when using 16 WAPs (part II).

51	52	53	54	55	56	57	58	59	60
3	3	3	3	3	3	3	3	3	3
3	**2**	**2**	3	3	3	**2**	3	**1**	**1**
0.3	0.3	0.1	0.1	0.1	0.1	0.1	0.3	0.4	0.4
0.2	0.4	0.5	0.4	0.2	0.4	0.5	0.3	0.3	0.3
0.5	0.3	0.4	0.5	0.7	0.5	0.4	0.4	0.3	0.3

Table 14: System predictions for area 1 when using 10 WAPs (part I).

1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
0.9	0.9	0.9	0.9	0.9	1.0	0.9	1.0	1.0	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0
0.1	0.1	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0

Table 15: System predictions for area 1 when using 10 WAPs (part II).

11	12	13	14	15	16	17	18	19	20
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
0.9	0.9	0.9	0.9	0.9	0.9	1.0	1.0	1.0	0.9
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.1	0.1	0.1	0.1	0.1	0.1	0.0	0.0	0.0	0.1

Table 16: System predictions for area 2 when using 10 WAPs (part I).

21	22	23	24	25	26	27	28	29	30
2	2	2	2	2	2	2	2	2	2
2	**1**	2	**1**	2	2	2	2	2	2
0.2	0.7	0.3	1.0	0.0	0.2	0.2	0.2	0.0	0.0
0.8	0.3	0.7	0.0	1.0	0.8	0.8	0.8	1.0	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 17: System predictions for area 2 when using 10 WAPs (part II).

31	32	33	34	35	36	37	38	39	40
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2
0.2	0.2	0.3	0.0	0.0	0.2	0.2	0.3	0.2	0.0
0.8	0.8	0.7	0.9	1.0	0.8	0.8	0.7	0.7	1.0
0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0

If we analyze the results for 5 access points (area 1) we can see that the system succeeds 19 times out of 20 and fails once out of 20 (fingerprint 11). For area 2 the system is wrong in its prediction for the fingerprint 39; it succeeds 19 times out of 20 and it fails once out of 20. As for area 3 it succeeds 14 times out of 20 and it fails 6 out of 20 times (fingerprints 46, 49, 51, 52, 56, and 57). In total the system succeeds 52 times out of 60 and

Table 18: System predictions for area 1 when using 5 WAPs (part I).

1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
0.8	0.6	0.7	0.8	0.7	0.8	0.6	0.8	0.5	0.5
0.0	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.2	0.1
0.2	0.2	0.1	0.0	0.1	0.2	0.4	0.2	0.3	0.4

Table 19: System predictions for area 1 when using 5 WAPs (part II).

11	12	13	14	15	16	17	18	19	20
1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1
0.3	0.4	0.6	0.7	0.7	0.7	0.8	0.5	0.4	0.5
0.0	0.2	0.1	0.2	0.2	0.0	0.0	0.0	0.2	0.2
0.7	0.4	0.3	0.1	0.1	0.3	0.2	0.5	0.4	0.3

Table 20: System predictions for area 2 when using 5 WAPs (part I).

21	22	23	24	25	26	27	28	29	30
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2
0.0	0.1	0.0	0.1	0.0	0.0	0.2	0.0	0.0	0.0
1.0	0.5	1.0	0.7	1.0	1.0	0.7	1.0	1.0	1.0
0.0	0.4	0.0	0.2	0.0	0.0	0.1	0.0	0.0	0.0

Table 21: System predictions for area 2 when using 5 WAPs (part II).

31	32	33	34	35	36	37	38	39	40
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	**1**	2
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.3	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0

Table 22: System predictions for area 3 when using 5 WAPs (part I).

41	42	43	44	45	46	47	48	49	50
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	**1**	3	3	**1**	3
0.3	0.2	0.1	0.0	0.1	0.6	0.4	0.3	0.5	0.3
0.0	0.1	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0
0.7	0.7	0.9	0.6	0.9	0.4	0.6	0.7	0.5	0.7

fails 8 times out of 60 which means there is a relative accuracy of 86.67%. If we compare this result with those obtained for 16 (80%) and 10 (63.34%) WAPs we get a difference of +6.67 and +23.33 percentage points, respectively. This finding is of special interest since it proves that when changes occur in the WAPs' measured RSSIs considered day to day we need to be stricter when selecting access points in order to obtain better predictions.

As we saw in the previous tables the system made, among others, the following wrong predictions: for area 1 (fingerprint 11), for area 2 (fingerprint 39), and for area 3 (fingerprint 52).

Table 23: System predictions for area 3 when using 5 WAPs (part II).

51	52	53	54	55	56	57	58	59	60
3	3	3	3	3	3	3	3	3	3
2	**1**	3	3	3	**1**	**1**	3	3	3
0.3	0.5	0.1	0.2	0.2	0.6	0.5	0.2	0.4	0.3
0.4	0.2	0.3	0.3	0.2	0.1	0.1	0.3	0.0	0.0
0.3	0.3	0.6	0.5	0.6	0.3	0.4	0.5	0.6	0.7

Table 24: System predictions for area 1 when using average fingerprint based on the number of WAPs.

tag(1)	pred (tag)	conf (1)	conf (2)	conf (3)
	16 WAPs			
11	1	0.5	0.3	0.2
10-11 (average)	1	0.5	0.2	0.3
10-11-12 (average)	1	0.5	0.2	0.3
9-10-11-12-13 (average)	1	0.5	0.2	0.3
	10 WAPs			
11	1	0.9	0.0	0.1
10-11 (average)	1	0.9	0.0	0.1
10-11-12 (average)	1	0.9	0.0	0.1
9-10-11-12-13 (average)	1	0.9	0.0	0.1
	5 WAPs			
11	**3**	0.3	0.0	0.7
10-11 (average)	1	0.8	0.0	0.2
10-11-12 (average)	1	0.4	0.3	0.3
9-10-11-12-13 (average)	1	0.5	0.1	0.4

There are the three scenarios in which we will introduce average fingerprinting to help the system make the right predictions for each of the areas. To do this we will calculate the average fingerprint using (a) the previous fingerprint, (b) the previous and subsequent ones, and (c) two previous and two subsequent fingerprints to the current one.

As we see in Table 24, when we use the average fingerprint resulting from these three methods (previous, previous and subsequent, two previous, and two subsequent) the system makes the right prediction with a reliability of an 80% percent, 40% percent, and 50% percent, respectively (Table 25).

For area "2" the system also makes the right prediction when it receives the average fingerprint in the three modalities described before. For 16 WAPs the system says that it belongs to area 2 with 80% confidence whereas for 5 WAPs this percentage increases up to 100% (Table 26).

For area 3, fingerprint 52, the system also makes the right prediction when it receives the average fingerprint for 5 and 16 WAPs scenarios with 50% and 80% percent of reliability, respectively. However, when the system takes 10 WAPs it still fails. Why? This is because whether we calculate the average fingerprint using the previous one (51), one previous fingerprint and subsequent one (51 and 53) or two previous and two subsequent fingerprints (50, 51, 53, and 54) the predictions made by the system for these fingerprints are also wrong so there is no way to get an upgrade.

Figur e 8: Validation points.

Tabl e 25: System predictions for area 2 when using average fingerprint based on the number of WAPs.

tag(2)	pred (tag)	conf (1)	conf (2)	conf (3)
	16 WAPs			
39	**3**	0.2	0.3	0.5
38-39 (average)	2	0.1	0.8	0.1
38-39-40 (average)	2	0.1	0.8	0.1
37-38-39-40-41 (average)	2	0.1	0.9	0.0
	10 WAPs			
39	2	0.2	0.7	0.1
38-39 (average)	2	0.0	0.9	0.1
38-39-40 (average)	2	0.0	1.0	0.0
37-38-39-40-41 (average)	2	0.3	0.7	0.0
	5 WAPs			
39	1	0.4	0.3	0.3
38-39 (average)	2	0.0	1.0	0.0
38-39-40 (average)	2	0.0	1.0	0.0
37-38-39-40-41 (average)	2	0.0	1.0	0.0

Tabl e 26: System predictions for area 3 when using average fingerprint based on the number of WAPs.

tag(3)	pred (tag)	conf (1)	conf (2)	conf (3)
	16 WAPs			
52	**2**	0.30	0.40	0.30
51-52 (average)	**2**	0.10	0.50	0.40
51-52-53 (average)	3	0.10	0.40	0.50
50-51-52-53-54 (average)	**2**	0.20	0.50	0.30
	10 WAPs			
52	**1**	0.90	0.10	0.00
51-52 (average)	**1**	0.90	0.10	0.00
51-52-53 (average)	**1**	0.90	0.10	0.00
50-51-52-53-54 (average)	**1**	0.90	0.10	0.00
	5 WAPs			
52	**1**	0.50	0.20	0.30
51-52 (average)	3	0.00	0.10	0.90
51-52-53 (average)	3	0.00	0.20	0.80
50-51-52-53-54 (average)	3	0.30	0.20	0.50

6.3.2. Results from the Second Experiment. A more complex scenario including 3 wall and door separated areas has also been implemented as presented in Figure 1. In order to validate the accuracy of the algorithms for estimating the area of an unlocated validation point based on the proposed training only with area border points, different validation points from those used for training were selected (randomly selected at each area) as shown in Figure 8.

A similar approach to evaluate the 15 machine learning algorithms as in the previous section was used. The same sampling schema as in Section 6.3.1 was used for the training points. The average results of the estimated areas for each validation point are shown in Table 27.

All validation points are properly located except point 15 which is a point very close to the border between areas 2 and 3. For a moving target to be located, the assessed trajectory information could be used to decide in which area a user is in order to improve the accuracy of the algorithms when approaching the border of two areas.

7. Conclusion

In this paper we have presented several ways to optimize fi gerprinting-based techniques for locating users indoors when only room level accuracy is required.

One of the parameters studied is the way to select the most convenient sampling points to measure training fingerprints in order to increase the accuracy while minimizing the memory and CPU execution requirements. Previous studies have focused on either an exhaustive grid of sample points

Table 27: Estimated areas for the validation points.

Validation points	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Estimated area	1	1	1	1	1	1	1	2	2	2	2	2	3	3	2	3	3	3

or a random selection of them. This paper concludes that, for room level accuracy, border points between rooms or areas contain the required information for the prediction algorithm in order to provide a similar or even better accuracy that the one provided by exhaustive or random sampling techniques.

The paper also provides a comparative analysis and evaluation of the performance of diff rent decision algorithms identifying under which circumstances each of them work better.

Another contribution presented in this paper is the proposal and evaluation of a new way to fi ter unstable access points in order to increase accuracy and minimize execution requirements. Taking into account the quality and stability over time of the received signals from the diff rent WAPs, the algorithm selects those maximizing both the signal strength and the minimal fluctuations over time. Several thresholds are defin d. Th experimental results show that the algorithms find optimal behavior when fi ters are configured to select between 10 and 16 WAPs.

Finally, the paper has presented that taking several consecutive fingerprints when locating a mobile device can improve the accuracy of the results. A compromise exists between the accuracy obtained and the time required to locate the user. The results show that 3 consecutive measures are a reasonable trade-off.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The research leading to these results has received funding from the "HERMES-SMART DRIVER" Project TIN2013-46801-C4-2-R within the Spanish "Plan Nacional de I+D+I" funded by the Spanish Ministerio de Economía y Competitividad, from the Grant PRX15/00036 for the "Estancias de Movilidad de Profesores e Investigadores Seniores en Centros Extranjeros de Enseñanza Superior e Investigación," from the Ministerio de Educación Cultura y Deporte, and from the Spanish Ministerio de Economía y Competitividad funded projects (cofinanced by the Fondo Europeo de Desarrollo Regional (FEDER)), IRENE (PT-2012-1036-370000), COMINN (IPT-2012-0883-430000), and REMEDISS (IPT-2012-0882-430000) within the INNPACTO program.

References

[1] I. Sabek, M. Youssef, and A. V. Vasilakos, "ACE: an accurate and effici t multi-entity device-free WLAN localization system," *IEEE Transactions on Mobile Computing*, vol. 14, no. 2, pp. 261–273, 2015.

[2] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *Computer*, vol. 34, no. 8, pp. 57–66, 2001.

[3] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 140–150, 2010.

[4] K. P. Subbu, C. Zhang, J. Luo, and A. V. Vasilakos, "Analysis and status quo of smartphone-based indoor localization systems," *IEEE Wireless Communications*, vol. 21, no. 4, pp. 106–112, 2014.

[5] A. Varshavsky, A. Lamarca, J. Hightower, and E. De Lara, "The skyLoc fl or localization system," in *Proceedings of the 5th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom '07)*, pp. 125–134, IEEE, White Plains, NY, USA, March 2007.

[6] Y. Jiang, X. Pan, K. Li et al., "ARIEL: automatic Wi-Fi based room fi gerprinting for indoor localization," in *Proceedings of the 14th International Conference on Ubiquitous Computing (UbiComp '12)*, pp. 441–450, Pittsburgh, Pa, USA, September 2012.

[7] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik, "Indoor localization without infrastructure using the acoustic background spectrum," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys '11)*, pp. 155–168, ACM, Washington, DC, USA, June-July 2011.

[8] K. P. Subbu, B. Gozick, and R. Dantu, "LocateMe: magnetic-fields-ba ed indoor localization using smartphones," *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 4, article 73, 2013.

[9] P. Bolliger, "Redpin—adaptive, zero-configur tion indoor localization through user collaboration," in *Proceedings of the 1st ACM International Workshop on Mobile Entity Localization and Tracking in GPS-Less Environments (MELT '08)*, pp. 55–60, San Francisco, Calif, USA, September 2008.

[10] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao, "A reliable and accurate indoor localization method using phone inertial sensors," in *Proceedings of the 14th International Conference on Ubiquitous Computing (UbiComp '12)*, pp. 421–430, Pittsburgh, Pa, USA, September 2012.

[11] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: zero-effort crowdsourcing for indoor localization," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (Mobicom '12)*, pp. 293–304, ACM, Istanbul, Turkey, August 2012.

[12] H. Shin, Y. Chon, Y. Kim, and H. Cha, "A participatory service platform for indoor location-based services," *IEEE Pervasive Computing*, vol. 14, no. 1, pp. 62–69, 2015.

[13] K. Subbu, C. Zhang, J. Luo, and A. Vasilakos, "Analysis and status quo of smartphone-based indoor localization systems," *IEEE Wireless Communications*, vol. 21, no. 4, pp. 106–112, 2014.

[14] H. Lin, Y. Zhang, M. Griss, and I. Landa, "WASP: an enhanced indoor locationing algorithm for a congested Wi-Fi environment," in *Mobile Entity Localization and Tracking in GPS-Less Environnments: Second International Workshop, MELT 2009, Orlando, FL, USA, September 30, 2009. Proceedings*, vol. 5801 of *Lecture Notes in Computer Science*, pp. 183–196, Springer, Berlin, Germany, 2009.

[15] M. Brunato and R. Battiti, "Statistical learning theory for location fi gerprinting in wireless LANs," *Computer Networks*, vol. 47, no. 6, pp. 825–845, 2005.

[16] A. Carlotto, M. Parodi, C. Bonamico, F. Lavagetto, and M. Valla, "Proximity classific tion for mobile devices using wi-fi environment similarity," in *Proceedings of the 1st ACM International Workshop on Mobile Entity Localization and Tracking in GPS-Less Environments (MELT '08)*, pp. 43–48, San Francisco, Calif, USA, September 2008.

[17] C. Zhou, H. Xie, and J. Shi, "Wi-Fi indoor location technology based on K-means algorithm," in *LISS 2014*, pp. 765–770, Springer, Berlin, Germany, 2015.

[18] K. Kaemarungsi and P. Krishnamurthy, "Modeling of indoor positioning systems based on location fingerprinting," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, vol. 2, pp. 1012–1022, March 2004.

[19] A. K. M. M. Hossain, H. N. Van, and W.-S. Soh, "Fingerprint-based location estimation with virtual access points," in *Proceedings of the 17th International Conference on Computer Communications and Networks (ICCCN '08)*, pp. 485–490, IEEE, August 2008.

[20] D. Lymberopoulos, J. Liu, X. Yang, R. R. Choudhury, V. Handziski, and S. Sen, "A realistic evaluation and comparison of indoor location technologies: experiences and lessons learned," in *Proceedings of the 14th International Conference on Information Processing in Sensor Networks (IPSN '15)*, pp. 178–189, ACM, Seattle, Wash, USA, April 2015.

[21] Y. Freund and L. Mason, "Th alternating decision tree learning algorithm," in *Proceedings of the 16th International Conference on Machine Learning*, pp. 124–133, Bled, Slovenia, June 1999.

[22] H. Shi, *Best-First Decision Tree Learning*, University of Waikato, Hamilton, New Zealand, 2007.

[23] J. Gama, "Functional trees," *Machine Learning*, vol. 55, no. 3, pp. 219–250, 2004.

[24] N. Landwehr, M. Hall, and E. Frank, "Logistic model trees," *Machine Learning*, vol. 59, no. 1-2, pp. 161–205, 2005.

[25] R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, Calif, USA, 1993.

[26] G. I. Webb, "Decision tree grafting from the all-tests-but-one partition," in *Proceedings of the 16th International Joint Conference on Artifici l Intelligence (IJCAI '99)*, vol. 2, Morgan Kaufmann Publishers, Stockholm, Sweden, July-August 1999.

[27] G. Holmes, B. Pfahringer, R. Kirkby, E. Frank, and M. Hall, "Multiclass alternating decision trees," in *Proceedings of the 13th European Conference on Machine Learning (ECML '02)*, pp. 161–172, Helsinki, Finland, August 2002.

[28] N. Landwehr, M. Hall, and E. Frank, "Logistic model trees," *Machine Learning*, vol. 59, no. 1, pp. 161–205, 2005.

[29] M. Sumner, E. Frank, and M. Hall, "Speeding up logistic model tree induction," in *Knowledge Discovery in Databases: PKDD 2005: 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, Porto, Portugal, October 3–7, 2005. Proceedings*, vol. 3721 of *Lecture Notes in Computer Science*, pp. 675–683, Springer, Berlin, Germany, 2005.

[30] R. Kohavi, "Scaling up the accuracy of naive-bayes classifie s: a decision-tree hybrid," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 202–207, Portland, Ore, USA, August 1996.

[31] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[32] P. Bolliger, Redpin, 2008, http://www.redpin.org/.

[33] H. Lin, Y. Zhang, M. Griss, and I. Landa, "WASP: an enhanced indoor locationing algorithm for a congested Wi-Fi environment," in *Mobile Entity Localization and Tracking in GPS-Less Environnments*, pp. 183–196, Springer, Berlin, Germany, 2009.